

JAVA PROGRAMMING

REGIONAL KEY 2025

Page 1 of 3

```
1 //create AI Chatbots
2 //test market for which chatbots are most popular and which token rate is the most profitable
3
4 import java.io.FileNotFoundException;
5 import java.io.File;
6 import java.util.*;
7 import java.text.DecimalFormat;
8
9 public class AIChatBotGenerator_Regional
10 {
11
12     static Scanner sc = new Scanner(System.in); //SC1
13     static DecimalFormat df_Currency = new DecimalFormat(pattern:"#,##0.00"); //SC7
14     Run | Debug
15     public static void main (String args [])
16     {
17         ArrayList<Chatbots> bots = new ArrayList<Chatbots>(); //SC2
18         System.out.println(x:"How many test bots do you want to create?");
19         int numberOfBots = integerInputManager();
20
21         //SC3
22         String [] languages ={"English", "French", "Chinese", "Spanish", "Arabic", "Russian", "German", "Japanese", "Portuguese", "Hindi", "Italian", "Korean", "Turkish",
23                               "Vietnamese", "Indonesian", "Greek", "Hebrew", "Thai"};
24
25         bots = setChatbots(bots, languages, numberOfBots);
26         printChatbots(bots, df_Currency);
27     }
28     private static String[][] compatibleLanguagePairs = {
29         {"English", "Spanish"}, {"English", "French"}, {"English", "German"}, {"Japanese", "English"}, {"Japanese", "Hindi"}, {"Portuguese", "French"},
30         {"Spanish", "Portuguese"}, {"Spanish", "Italian"}, {"French", "Italian"}, {"Russian", "Chinese"}, {"Russian", "Korean"}, {"Russian", "Japanese"},
31         {"Russian", "German"}, {"Hindi", "Arabic"}, {"Japanese", "Korean"}, {"Russian", "Arabic"}, {"Spanish", "Arabic"}, {"Russian", "French"},
32         {"Chinese", "Korean"}, {"Japanese", "Vietnamese"}, {"Arabic", "Hebrew"}, {"Greek", "Hebrew"}, {"Arabic", "Indonesian"}, {"Russian", "Indonesian"},
33         {"Chinese", "Thai"}, {"Japanese", "Vietnamese"}, {"English", "Chinese"}, {"Japanese", "Thai"}, {"Indonesian", "Thai"}, {"Chinese", "Spanish"},
34         {"Arabic", "Turkish"}, {"Greek", "Turkish"}, {"Vietnamese", "Thai"}, {"Vietnamese", "French"}, {"Vietnamese", "Indonesian"}, {"Turkish", "Spanish"},
35     };
36
37     //This will create the random bots
38     private static ArrayList<Chatbots> setChatbots(ArrayList<Chatbots> list_of_Bots, String[] languages, int noB) {
39         Random rand = new Random();
40         Chatbots chatBots;
41         ArrayList<Chatbots> bots = list_of_Bots;
42         DecimalFormat df = new DecimalFormat(pattern:"#.0000");
43
44         int numberOfBots = noB;
45         double tokenRate;
46         int complexity;
47         String primaryLang;
48         String secondaryLang;
49         for (int i = 0; i < numberOfBots; i++) {
50             do {
51                 primaryLang = languages[rand.nextInt(languages.length)];
52                 secondaryLang = languages[rand.nextInt(languages.length)];
53             } while (primaryLang.equals(secondaryLang));
54
55             complexity = rand.nextInt(bound:16) + 1;
56             double tempTokenRate = rand.nextDouble() * 7.5 + 0.1;
57             String tempString = df.format(tempTokenRate);
58             tokenRate = Double.parseDouble(tempString);
59             chatBots = new Chatbots(primaryLang, secondaryLang, complexity, tokenRate);
60             bots.add(chatBots);
61         }
62         return bots;
63     }
64     private static boolean isCompatible(String primaryLang, String secondaryLang) { //SC15
65         for (String[] pair : compatibleLanguagePairs) {
66             if ((pair[0].equals(primaryLang) && pair[1].equals(secondaryLang)) ||
67                 (pair[1].equals(primaryLang) && pair[0].equals(secondaryLang))) {
68                 return true;
69             }
70         }
71         return false;
72     }
73 }
```

```

74 private static void printChatbots(ArrayList<Chatbots> bots, DecimalFormat df_Currency) {
75     System.out.println(x:"-----");
76     System.out.printf(format:"| %-4s | %-15s | %-15s | %-20s | %-10s | %-15s | %-14s | \n", //SC13
77         ...args:"No.", "Primary Lang.", "Secondary Lang.", "Algorithm Complexity", "TokenRate", "Estimated Cost", "Compatibility");
78     System.out.println(x:"-----");
79     int i = 1;
80     for (Chatbots b : bots) {
81         String compatibility = isCompatible(b.getPrimaryLanguage(), b.getSecondaryLanguage()) ? "Compatible" : "Not Compatible";
82         System.out.printf(format:"| %-4d | %-15s | %-15s | %-20d | %-10.2f | $%-14s | %-14s | \n", //SC14
83             i, b.getPrimaryLanguage(), b.getSecondaryLanguage(), b.getComplexity(), b.getTokenRate(), df_Currency.format(b.getEstimatedCost()), compatibility);
84         i++;
85     }
86     System.out.println(x:"-----");
87 }
88
89 private static int integerInputManager()
90 {
91     int temp;
92     while(true){
93         try{ //SC4
94             do{
95                 System.out.print(s:"Please enter in a value between 5 and 20: ");
96                 temp = sc.nextInt();
97                 if (temp > 20 || temp < 5) //SC5
98                     System.out.println(x:"Your entry is out of range.\n");
99                 sc.nextLine();
100             }while(temp > 20 || temp < 5); //SC5
101             return temp;
102         }
103         catch(InputMismatchException e) //SC4
104         {
105             sc.next();
106             System.out.println(x:"\nPlease enter a correct value.");
107         }
108     }
109 }
110
111 }
112
113
114 }
115 //Given
116 //////////////////////////////////////////////////
117 class Chatbots
118 {
119     String primaryLang;
120     String secondaryLang;
121     int complexity_Level;
122     double tokenRate;
123     double estimatedCost = 0;
124
125     public Chatbots()
126     {
127         primaryLang = "ASCII";
128         secondaryLang = "ASCII";
129         complexity_Level = 0;
130         tokenRate = 0.0;
131     }
132
133     public Chatbots(String primaryLang, String secondaryLang, int complexity_Level, double tokenRate)
134     {
135         this.primaryLang = primaryLang;
136         this.secondaryLang = secondaryLang;
137         this.complexity_Level = complexity_Level;
138         this.tokenRate = tokenRate;
139         setEstimatedCost();
140     }
141
142

```

```
143     public String getPrimaryLanguage()
144     {
145         | return primaryLang;
146     }
147
148     public String getSecondaryLanguage()
149     {
150         | return secondaryLang;
151     }
152
153     public double getTokenRate()
154     {
155         | return tokenRate;
156     }
157
158     public int getComplexity()
159     {
160         | return complexity_Level;
161     }
162
163     public void setEstimatedCost()
164     {
165         | estimatedCost = getComplexity() * getTokenRate();
166     }
167
168     public double getEstimatedCost()
169     {
170         | return estimatedCost;
171     }
172
173 }
```